Theodor H. Nelson, October 1968

Final Report

STUDY FOR THE DEVELOPMENT OF HUMAN INTELLECT AUGMENTATION TECHNIQUES

By: D. C. ENGELBART

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
MAIL STOP 126
LANGLEY STATION
HAMPTON, VIRGINIA 23365

CONTRACT NAS1-5904

STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA

July 1968

Final Report

STUDY FOR THE DEVELOPMENT OF HUMAN INTELLECT AUGMENTATION TECHNIQUES

By: D. C. ENGELBART

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
MAIL STOP 126
LANGLEY STATION
HAMPTON, VIRGINIA 23365

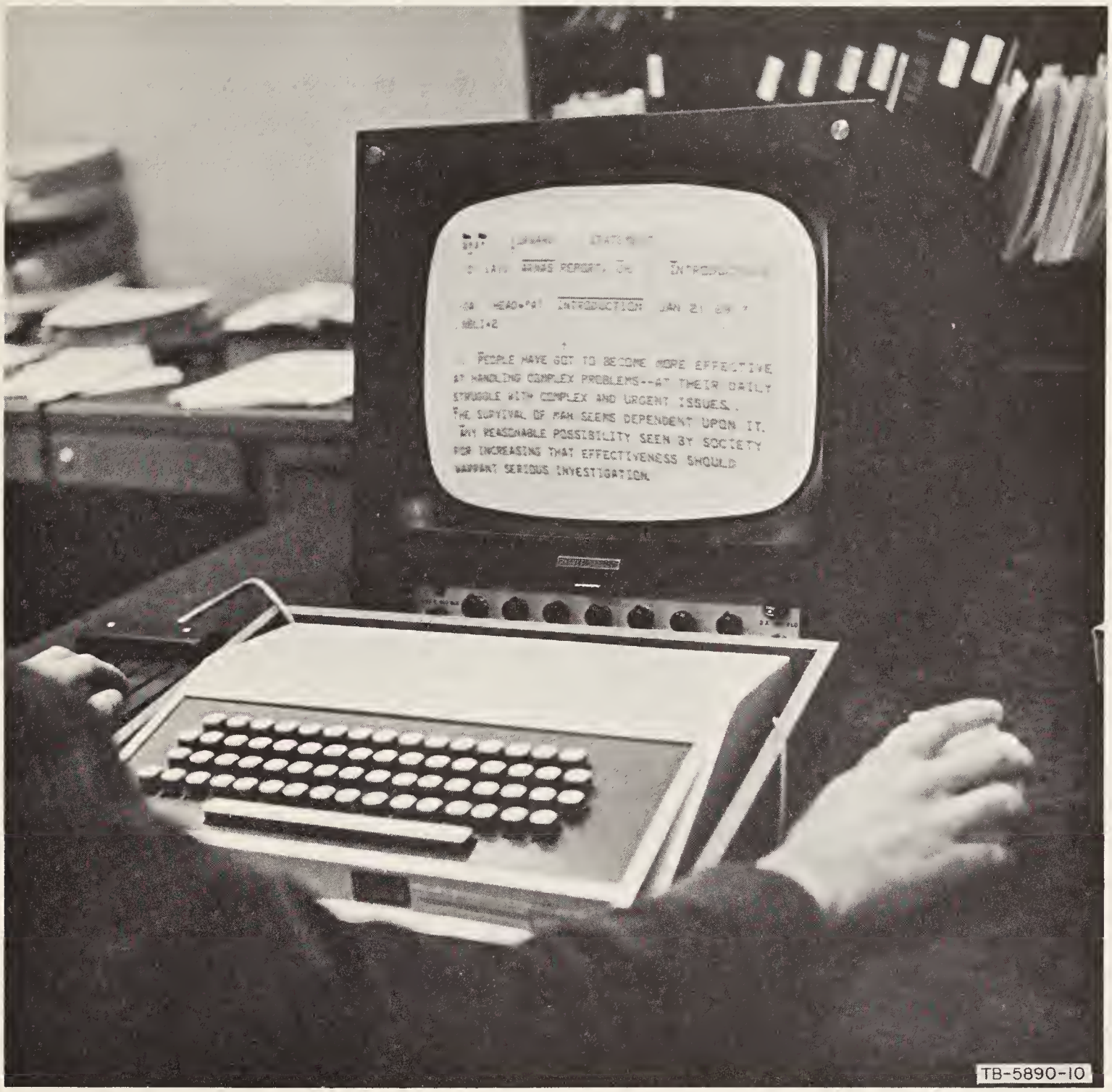
CONTRACT NAS1-5904

SRI Project 5890

Approved: BONNAR COX, ACTING MANAGER
Systems Engineering Laboratory

TORBEN MEISLING, EXECUTIVE DIRECTOR
Information Science and Engineering

Copy No. 58



USER'S WORK-STATION CONSOLE

ABSTRACT

This report covers a two-year project, at the ninth year of a growing, multiproject program that is exploring the value of computer aids to augmenting human intellectual capability. Outlined briefly are the background and the "bootstrapping" nature of the program (e.g., the report was produced with the experimental computer aids) and its status at the start of this project. Advances during the project were in the programming system (CRT on-line debugging, etc.), in both off-line and on-line computer-aided text-manipulation systems implemeoted on a CDC 3100, and in specifications, designs, and nearly completed implementation of a time-shared system on an SDS 940, which will be committed to serving 12 on-line CRT consoles for this program. Special features of the design include commercial-TV display consoles, special equipment to reduce CPU overhead to a few percent for servicing I/O, a new machine-oriented, systems programmer compiler language (MOL940), a compiler-compiler (Tree Meta), and a Control Metalanguage Translator for compiling the interactive part of the on-line system from a special Control Metalanguage description. Experiences with usage to date lead to an assortment of future research possibilities for both the computer aids and their utilization. A "Bootstrap Community" provides the planning basis for the choice and character of future research activities.

FOREWORD

This is the final report of a two-year project at Stanford Research Institute in man-computer interaction. The project has been jointly sponsored by NASA and ARPA, and provides the nucleus for a continuing, multi-sponsor program organized within the Augmented Human Intellect (AHI) Research Center. The Center's research program has been building up at SRI since 1959, via a series of coordinated projects and facility-development stages.

One of the experimental activities of the project is to keep its working records in computer storage, and to do as much as possible of the daily manipulation with real-time interactive computer aids. As an example, except for diagrams and photographs, this entire report has been carried through all of its stages of composition, modification, and production by such means (i.e. through the automatic preparation of reproduction copy).

CONTENTS

ILLUSTRATIONS

SUMMARY

Chapter I of this report is a general introduction to the activities of the AHI Research Center at Stanford Research Institute and to the facilities in use for the research. Chapter II describes the strategy of research and the experimental environment in the Center, entailing a "bootstrapping" concept and complex systems for on-line, interactive computer aid to intellect on a daily, full-time basis.

Chapter III describes the development of user systems -- the aspects of the computer systems that are apparent and useful to the user; Chapter IV deals with special system-design techniques which have evolved in the implementation of user systems.

Chapter V is a discussion of the results that have been observed from intensive usage of the systems, in terms of possibilities for human intellect augmentation. Chapter VI presents conclusions and recommendations.

I INTRODUCTION

1 In the Augmented Human Intellect (AHI) Research Center at Stanford Research Institute a group of researchers is developing an experimental laboratory around an interactive, multiconsole computer-display system, and is working to learn the principles by which interactive computer aids can augment the intellectual capability of the subjects.

2 For the first part of the project, the Center's experimental facility consisted of a CDC 3100 computer that was equipped with one CRT console.

2A The basic strategy of the AHI research program is to use its own augmentation-system developments for its own work. Earlier AHI-Center projects had developed an interactive system (see English et al., 1965; English et al., 1967; and Engelbart, 1967), and this was improved considerably during the first six months. Although we developed some very effective working aids on this facility, it became apparent that effective utilization of on-line aids requires considerably more console-service accessibility than this facility provided. This limited accessibility handicapped our research approach of building, using, and evolving our own augmentation system.

2B Relatively early in the contract period, this limited-access handicap received enough recognition to bring about a readjusted level of support (from ARPA, channelled through an RADC contract) and a shift in our activity. For the last 18 months the main activity has been the planning and implementation of a multiconsole system dedicated to the AHI Center's "bootstrapping" activity.

3 For its laboratory facility, the Center now has the following:

3A An SDS940 time-sharing computer (65K, 24-bit core) with a 4.5 megabyte swapping drum and a 96 megabyte file-storage disk. This simultaneously serves 12 CRT work stations.

3B The display in a user's office (see Frontispiece) or workroom appears on a commercial television monitor (high resolution -- 875 lines) and provides both character and vector portrayals. An SRI mouse (cursor device) allows screen pointing and selection, and a keyboard is used for entering both data and control information.

3C The user features being first developed for this facility are essentially the same as on the prototype 3100 system. (See the section below on "User Systems".)

3D There are several notable features in the design of this experimental facility.

3D1 Specially developed hardware enables the I/O for these highly interactive CRT work stations to be carried out with almost no

overhead on the central processor.

3D2 The display system uses a unique mix of conventional, refreshed, random-deflection CRTs and commercial television equipment to obtain overall economy, black-on-white displays, and the ability to superimpose different TV images on one display.

3D3 Special-purpose high-level languages and associated compilers provide rapid, flexible development and modification of the repertoire of service functions and of their control procedures (the latter being the detailed user actions and computer feedback involved in controlling the application of these service functions).

3D4 Developed under an earlier project to facilitate these controlling actions are the SRI mouse (a hand-held X-Y transducer usable on any flat surface) and a five-key handset for one-handed entry of control codes and literal input.

4 User files are organized in hierarchical structures of data entities, each composed of arbitrary combinations of text and figures. A large repertoire of coordinated service features enables a skilled user to compose, study, and modify these files with great speed and flexibiliy, and to have searches, analyses, data manipulation, etc. done for him. In particular, special sets of conventions, functions, and working methods have been developed to aid programming, documenting, retrieval, project management, group interaction, logical design, and hard-copy production.

5 Before describing in more detail the design of the hardware and software for these systems, this report discusses the purpose and approach for our coordinated "bootstrap" program, where our emphasis is on learning how to derive maximum value to human effectiveness by harnessing computer aids.

5A Our primary research concern is actually with such things as the repertoire of service functions provided the user, the means he uses to control the application of these functions, his changes in working conventions and procedures, the new ways he and others can now work together as a team, etc. These primary research factors are what determine the secondary goals of developing the instrumentation for our research (i.e., the hardware and software).

II RESEARCH STRATEGY AND ENVIRONMENT

I Strategy

IA The research objective is to develop principles and techniques for designing an "augmentation system."

IA1 This includes concern not only for the technology of providing interactive computer service, but also for changes in ways of conceptualizing, visualizing, and organizing material, as well as procedures and methods for working individually and cooperatively.

IB The research approach is strongly empirical: at the workplace of each member of the group, we aim to provide a nearly full-time CRT work station, and then endeavor continuously to improve the services at the stations, thus making them more valuable to group members in all roles and activities.

IB1 Among the research activities of the group are the following: evolutionary development of a complex hardware-software system, design of new task procedures for the system's users, and careful documentation of the evolving system designs and user procedures.

IB2 The group also has the usual tasks of managing its activities, keeping up with outside developments, publishing reports, etc.

IB3 Hence, the particulars of the augmentation system evolving here will reflect the nature of these tasks; i.e., the system is aimed at augmenting a system-development project team. Though the primary research goal is to develop principles of analysis and design for augmentation of human capability, choosing this research approach yields the valuable secondary benefit of a system tailored to help develop complex computer-based systems.

IC Having the staff of the AHI Center serve both as the research group and as the subject group produces a unique twist to this approach.

IC1 This "bootstrap" group has the interesting (recursive) assignment of developing tools and techniques to make it more effective at carrying out its assignment.

IC2 Its tangible product is a constantly improving augmentation system for use in developing and studying augmentation systems.

IC3 This system can be transferred -- as a whole or by pieces of concept, principle and technique -- to help others develop augmentation systems for many other disciplines and activities.

1D We are concentrating fully upon becoming able to do all of our daily work on line, so that we can:

1D1 Put into computer store all of our specifications, plans, designs, programs, documentation, reports, memos, bibliography, reference notes, etc.

1D2 Do all our scratch work, planning, designing, debugging, etc., and a good deal of our internal communication, via the consoles.

1D3 Maximize the coverage of our documentation, using it as a dynamic and plastic structure that we continually develop and alter to represent the current state of our evolving goals, plans, designs, procedures, and data.

1E The display-computer system to support this experiment is just becoming operational, supplied with a basic display-oriented user system that we have evolved over five years and through three other computers. Below are described the principle features of these systems.

2 Single-Console System

2A When the project began, we had just replaced our earlier CDC 160A with the present CDC 3100. The 160A facility and the on-line system implemented with it are described in English et al., 1965.

2B The CDC 3100 computer has the following configuration:

2B1 Memory: 16,000 words, 24 bit, 1.75 microseconds

2B2 Three I/O channels, one of which is compatible with the interface previously used on some of our equipment for coupling to the CDC 160A

2B3 Paper-tape I/O

2B4 Three magnetic-tape transports

2B5 One IBM 1311 disk file (2,000,000 character capacity)

2B6 One 150-line/minute printer

2B7 Card reader, 1200 cards per minute.

2C Work Station

2C1 Special (SRI) interface equipment couples the 3100 to a

display generator and to the various keyboards and selection devices of the work station.

2C2 The display is driven by a character generator which can generate vectors and a repertoire of 63 characters at about 100,000 characters per second.

2D On-Line Assembling and Debugging (COPE)

2D1 COPE (Controlled On-Line Program Executer) is an integrated machine-language programming system for creating, modifying, debugging, and filing programs. Its functions include those normally assigned to distinct programming systems such as assemblers, debuggers, and file-monitor systems. (See Hopper, 1968, for a detailed description).

2D2 COPE was finished early in 1966 and has become the basic operating system for our on-line-system development on the 3100. Its on-line (CRT) debugging facility makes it very valuable to us, especially when combined with on-line editing of the source code as provided by our NLTS (see below).

2D3 A FORTRAN program was written which translates COMPASS (the CDC-supported assembler system for the 3100) source code into COPE source code -- except for the macros. With the help of this program, we mapped the implementation of the COPE system into the COPE language, so that its subsequent debugging and extensions have been done with the aid of its own on-line debugging features.

2D4 COPE is able to load and link to programs written in 3100 FORTRAN IV or in COMPASS.

2D4A FORTRAN and COMPASS routines, after translation into relocatable machine code by standard CDC translators, can be loaded by the COPE monitor, at the same time that it is assembling regular COPE code, so that processes written in these other languages can be mixed with COPE processes.

2D5 Also, flexible means to overlay (from disc) are provided.

2D5A A COPE program may be written with parts of it designated to be stored on the disk and "overlaid" onto specified sections of core when called by another part of the program.

2D5B This allows free growth of the operating systems, so that we can develop large repertoires of commands and/or more sophisticated operations.

3 SDS 940 Multiconsole System

3A In addition to the SDS 940, the facility includes peripheral equipment made by other manufacturers as well as equipment designed and constructed at SRI.

3B All of the non-SDS equipment is interfaced through the special devices channel which connects to the second memory buss through the SDS memory interface connection (MIC).

3B1 This equipment together with the random-access disc (RAD) is a significant load on the second memory buss. Not including the proposed "special operations" equipment, the maximum expected data rate is approximately 264,000 words per second or one out of every 2.1 memory cycles. However, with the 940 variable priority scheme for memory access (Pirtle, 1967), we expect less than 1 percent degradation in CPU efficiency due to this load.

3B2 This channel and controllers (with the exception of the disc controller) was designed and constructed at SRI.

3B2A With the 940 time-sharing system, we suspect that response will be limited by the CPU time available to the users for processing. In the design of the hardware serving the work stations we have attempted to minimize the CPU time by making the system as automatic as possible in its access to memory, and by formatting the data in memory so as to minimize the executive time necessary to process it for the users.

3C Figure 1 is a block diagram of the special-devices channel and associated equipment. The major components are as follows:

3C1 Executive Hardware

3C1A This is essentially a multiplexer that allows independent, asynchronous access to core from any of the 6 controllers connected to it. Its functions are as follows:

3C1A1 Decoding certain instructions from the computer and passing them along as signals to the controllers

3C1A2 Accepting addresses and requests for memory access (input or output) from the controllers, determining relative priority among the controllers, synchronizing to the computer clock, and passing the requests along to memory via the MIC.

3C1B The executive hardware includes a debugging panel that allows any of the 6 controllers to be operated "off-line" without interfering with the operation of other controllers.

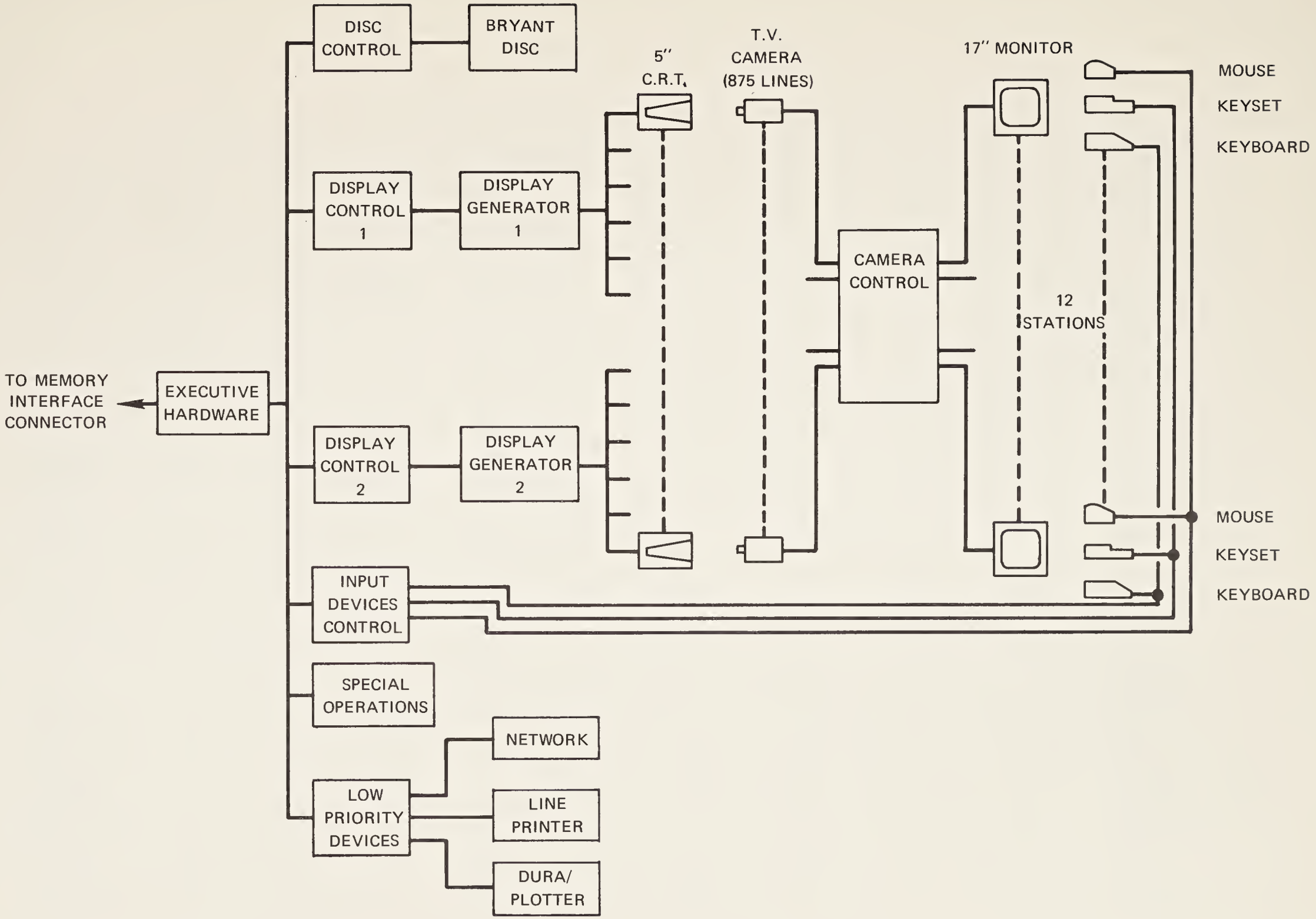


FIG. 1 SPECIAL–DEVICES CHANNEL

3C2 Disc File

3C2A This is a Model 406! Bryant disc, selected for compatibility with the continued 940-system development by Berkeley's Project GENIE, where extensive file-handling software was developed.

3C2B As formatted for our use, the disc will have a storage capacity of approximately 32 million words, with a data-transfer rate of 40,000 words per second and average access time of 85 milliseconds.

3C2C The disc controller is being designed by Bryant in close cooperation with SRI and Project GENIE.

3C3 Display System

3C3A The display system consists of two identical subsystems, each with display controller, display generator, 6 CRT's, and 6 closed-circuit television systems.

3C3B The display controllers process display-command tables and display lists that are resident in core, and pass along display-buffer contents to the display generators.

3C3C The display generators and CRT's were developed commercially to our specifications. Each has general character-vector plotting capability, and will drive six 5-inch high-resolution CRT monitors on which the display pictures are produced. Each generator will accept display buffers consisting of instructions (beam motion, character writing, etc.) from the controller.

3C3C! Character writing time is approximately 8 microseconds, allowing an average of !000 characters on each of the six monitors when regenerating at 20 cps.

3C3D A high-resolution (875-line) closed-circuit television system transmits display pictures from each CRT to a television monitor at the corresponding work-station console.

3C3E This system was developed as a "best solution" to our experimental-laboratory needs. However, it turned out to have valuable properties for more widespread use:

3C3E! Since only all-black or all-white signal levels are being treated, the scan-beam current on the cameras can be reduced to achieve a short-term image-storage effect that yields flicker-free TV output even when the refresh rate is

as low as 15 cps. This allows a display generator to sustain about four times as much material as could be sustained with direct-view refresh tubes.

3C3E2 The total cost of small CRT, TV camera, amplifier-controller, and monitor came to about $5500 per work station -- where a similar sized random-deflection, display-quality CRT would cost around $10,000 and would be harder to drive remotely.

3C3E3 Another important cost feature in some system environments favors this TV approach. The expensive part is centrally located; each outlying monitor costs only about $600, so terminals can be set up even where they are not used very often, enabling some video recoupling in the central establishment to take one terminal down and put another up.

3C3E4 An interesting feature of the video system is that with the flick of a switch the video signal can be inverted, so that the image picked up as bright lines on dim background may be viewed as black lines on a light background. There is a definite user preference for this inverted form of display.

3C4 Input-Device Controller

3C4A In addition to the television monitor, each work-station console will have a keyboard, binary keyset, and mouse.

3C4B The controller reads the state of these devices at a preset interval (about 30 milliseconds) and writes it into a fixed location table in core.

3C4B1 Bits are added to information from the keyboards, keysets and mouse switches to indicate when a new character has been received or a switch has changed state during the sample period. If there is a new character or switch change, an interrupt is issued after the sample period.

3C4B2 The mouse coordinates are formatted as a beam-positioning instruction to the display generator. Provisions are made in the display controller for including an entry in the mouse-position table as a display buffer. This allows the mouse position to be displayed continuously without any attention from the CPU.

3C5 Special Operations

3C5A The box with this label in Fig. 1 is at this time only a provision in the executive hardware for the addition of a high-speed device. We have tentative plans for adding special hardware to provide operations not available in the 940 instruction set, such as character-string moves and string-pattern matching.

3C6 Low-Priority Devices

3C6A This controller accommodates three devices with relatively low data-transfer rates. At this time only the line printer is implemented, with provisions for adding an on-line typewriter (Dura), a plotter, and a terminal for the proposed ARPA computer network.

III DEVELOPMENTS IN USER-SYSTEM FEATURES

I On-Line User Systems

IA Introduction

IA1 The user features outlined below are essentially common to both facilities, but the particulars are given for the new multiconsole system.

IA2 The basic facility will have the following characteristics:

IA2A I2 CRT consoles, of which usually I0 will be located in offices of AHI research staff. Each console may operate entirely independent of the others.

IA2B Each individual has private file space, and the group has community space also, on a high-speed disc with a capacity of 96 million characters.

IA2C The system is not intended to serve a general community of time-sharing users, but is being shaped in its entire design toward the special needs of the "bootstrapping" experiment.

IB Work Stations

IB1 The display at each of the work stations (see Frontispiece) is provided on a high-resolution, closed-circuit television monitor.

IB1A The alphanumeric keyboard has 94 normal characters in two cases. A third-case shift key provides for future expansion, and five special keys are used for system control.

IB2 Cursor Device (Mouse)

IB2A The mouse (see Fig. 2) produces two analog voltages as the two wheels rotate, each changing in proportion to the X or Y movement over the table top.

IB2A1 These voltages control -- via an A/D converter, the computer's memory, and the display generator -- the coordinates of a tracking spot with which the user may "point" to positions on the screen.

IB2A2 Three buttons on top of the mouse are used for special control.

IB2B A set of experiments comparing (within our techniques of interaction) the relative speed and accuracy obtained with this and other selection devices showed the mouse to be better than

FIG. 2 UNDERSIDE OF CURSOR DEVICE (MOUSE)

a light pen or a joy stick (English et al., 1963; English et al., 1967).

1B2B1 Compared to a light pen, the mouse is generally less awkward and fatiguing to use, and it has a decided advantage for use with scan-raster, write-through storage tube, projection, or multiviewer display systems.

1B3 Five-Key Handset

1B3A The five-key handset has 31 chords or unique key-stroke combinations. In one of the five "cases," these are interpreted as lower-case letters and punctuation. In another, the letters are interpreted as upper case. (The keystroke codes for the letters A to Z are the binary combinations for 1 to 26.) Two other cases contain digits and special characters; the fifth case is "control case."

1B3A1 A particular chord (the same chord in each case) will always transfer subsequent input-chord interpretations to control case.

1B3A2 In control case, one can delete through recent input, specify underlining for subsequent input, transfer to another case, visit another case for one character or one word, etc.

1B3B One-handed typing with the handset is slower than two-handed typing with the standard keyboard. However, working with one hand on the handset and one on the mouse, the coordinated interspersion of control characters and short literal strings from one hand with mouse-control actions by the other yields considerable advantage in speed and smoothness of operation.

1B3B1 For literal strings longer than about ten characters, one tends to transer from the handset to the normal keyboard.

1B3B2 Both from general experience and specific experiment, it seems that enough handset skill to have its presence begin to pay off can generally be achieved with about five hours' practice -- beyond this, skill grows with usage.

1C Structure of Files

1C1 Our working information is organized into files, with flexible means for setting up indices and directories and for moving files in and out of storage. Each file is itself highly

structured in its internal organization.

|C|A The specific structure of a given file is determined by the user, and is an important part of his conceptual and study-manipulate treatment of the file.

|C2 The introduction of explicit "structuring" to our working information stems from a very basic feature of our conceptual framework (Engelbart, 1963) regarding means for augmenting human intellect.

|C2A With the view that the symbols one works with are supposed to represent a mapping of one's associated concepts, and further that one's concepts exist in a "network" of relationships as opposed to the essentially linear form of actual printed records, it was decided that the concept-manipulation aids derivable from real-time computer support could be appreciably enhanced by structuring conventions that would make explicit (for both the user and the computer) the various types of network relationships among concepts.

|C2B As an experiment with this concept, we adopted some years ago the convention of organizing all information into hierarchical structures, with provisions for arbitrary cross-referencing among the elements of a hierarchy.

|C2B| The principal manifestation of this hierarchical structure is the breaking up of text into arbitrary segments called "statements," each of which bears a number showing its serial location in the text and its "level" in an "outline" of the text.

|C2C To set up a reference link from Statement A to Statement B, one may refer in Statement A either to the location number of B or to the "name" of B. The difference is that the number is vulnerable to subsequent structural change, whereas the name stays with the statement through changes in the structure around it.

|C2C| The convention is that the first word of a statement will be treated as the name for that statement, if it is enclosed in parentheses. For instance, Statement 0 on the screen of the Frontispiece is named "AFI". References to these names may be embedded anywhere in other statements, for instance as "see(spec)," with special format allowing a viewer to know explicitly that this refers to a statement named "spec". The reference may also appear as a string of characters in a context such that the viewer can infer the

referencing.

!C2C2 This naming and linking, when added to the basic hierarchical form, yields a general structuring capability that is quite flexible. These structuring conventions are expected to evolve relatively rapidly as our research progresses.

!C3 For some material, the structured statement form may be undesirable. In these cases, there are means for suppressing the special formatting in the final printout of the structured text.

!C4 The basic validity of this approach has been well established by our subsequent experience.

!C4A We have found that in both off-line and on-line computer aids, the conception, stipulation, and execution of significant manipulations are made much easier by the structuring conventions.

!C4B Also, in working on line at a CRT console, not only is manipulation made much easier and more powerful by the structure, but a user's ability to get about very quickly within his data and to have special "views" of it generated to suit his need are significantly aided by the structure.

!C5 We have come to write all of our documentation, notes, reports, and proposals according to these conventions, because of the resulting increase in our ability to study and manipulate them during composition, modification, and usage. Our programming systems also incorporate the conventions. We have found it to be fairly universal that after an initial period of negative reaction in reading explicitly structured material, one comes to prefer it to material printed in the normal form.

!D Studying Files

!D! We use computer aids for two principal "studying" operations: (1) locating and positioning the user's view to particular passages whithin a file, and (2) constructing special views of the material beginning with that passage.

!D!A In normal, off-line document studying, a person often does the first type of operation, but the second is like a scissors-and-staple job and is rarely done just to aid one's studying.

!D!B (A third type of service operation that will undoubtedly be a significant aid to studying is question answering. We do

not have this type of service.)

!D2 Controlling Location of Displayed "View"

!D2A Designating a "point" in the file to which one wants to "go", i.e. at which he wants the construction of his next display view to begin, is usually done by specifying a given "view-start" statement within the file. Such a designation may be made in several ways:

!D2A! By direct selection when the statement is visible on the display -- the statement (actually any character within the statement) is simply pointed to with the mouse.

!D2A2 By indirect "marker" selection -- where the statement is not on the display but the user has previously placed a temporary, named marker on some character of the statement, he can select that character (and thus the statement) by giving the name of the marker. Our control procedures make this essentially as quick and simple as a direct-screen selection.

!D2A3 By furnishing either the name or the location number of the statement, which can be done in either of two basic ways:

!D2A3A Typing it in from the keyboard

!D2A3B Selecting a word from the file that is the name or number (or contains it within the special reference-link word format) -- and this may be done either directly or via an indirect marker selection.

!D2B Specifying where to go relative to a designated statement -- a user may request that he be taken to a particular statement bearing a specified structural relationship to the one specifically identified. For instance, after identifying Statement 3E4 by one of the above means (assume it to be a member of the list 3E! through 3E7), a user may ask to be taken to its:

!D2B! Successor, i.e. Statement 3E5

!D2B2 Predecessor, i.e. Statement 3E3

!D2B3 List Tail, i.e. Statemnt 3E7

!D2B4 List Head, i.e. Statement 3E!

|D2B5 List Source, i.e. Statement 3E

|D2B6 Sub-head, i.e. Statement 3E4A.

|D2C Besides being taken to an explicitly identified statement, a user may ask to go to the first statement in th file (or the next after the current location) that contains a specified word or string of characters.

|D2C| He may specify the search string either by direct screen selection, or by indirect marker selection.

|D3 Controlling Form of View

|D3A The "normal" view beginning at a given location resembles a frame cut out from a long scroll upon which the statements are printed in sequential order. Such a view is displayed in the Frontispiece.

|D3B Otherwise we have three independently variable view-specification conditions that may applied to the construction of the displayed view: level clipping, line truncation, and content filtering.

|D3B| Level and Truncation Specifications

|D3B|A Given a specified level parameter, L (L=1,2,....,ALL), the view generator will first put on the display the designated view-start statement, and thereafter proceed as for the normal case except skipping over all statements whose "level" in the hierarchy is greater than L. (For example, Statement 3E4 is third level, 3E second, 4B2C| fifth, etc.)

|D3B|A| Thus to see only first-, second-, and third-level statements, one specifies L=3.

|D3B|B Given a specified truncation parameter, T (T=1,2,...,ALL), the view generator will display only the first T lines of each statement.

|D3B2 Content Specification

|D3B2A Given specifications for desired content (written in a special high-level content-analysis language), the view generator will display only those statements that have the desired content. One can test for the existence of simple strings, or logical combinations thereof, or for such things as having the word "memory" within four

words of the word "allocation".

!D3C "Freezing" Statements

!D3C! One may also preempt an arbitrary amount of the upper portion of the screen for holding a collection of "frozen" statements. The remaining lower portion is treated as a reduced-size scanning frame, and the view generator follows the same rules for filling it as described above.

!D3C!A The frozen statements may be independently chosen or dismissed; each may have line truncation independent of the rest; and the order in which they are displayed is arbitrary and readily changed. Any screen-select operand for any command may be selected from any portion of the display.

!D3D Figures 3 and 4 show views generated from the same starting point with different level-clipping parameters. This example happens to be of a program written in our Machine-Oriented language (MOL, see below).

!D3E Figure 5, showing a view of a program written in our Control Metalanguage (CML, see below), demonstrates the freezing feature, where Statements 4C, 4C2, 3B, 3B!, 3B2, 3B3, and 3B4 are frozen, and Statements from 3J on are shown normally with L = 3, T = !.

!D3E! The freezing here was used to hold for simultaneous view four different, functionally related process descriptions; the subroutines (named +bug!spec and +wait) were located each by use of the "hop-to-name" feature described above.

!E Modifying Files

!E! Here we use a standard set of operations, specifying with each operation a particular type of data entity. (The upper-case letters below are the single-letter designators used for specifying operations and entities.):

!E!A Operations: Delete, Insert, Replace, Move, Copy

!E!B Entities (for structure manipulation): Statement, Branch (statement plus all substructure), Group (sublist of branches), or Plex (complete list of branches)

!E!C Entities (within statements): Character, Text (arbitrary string), Word, Visible (print string), Invisible (gap string).

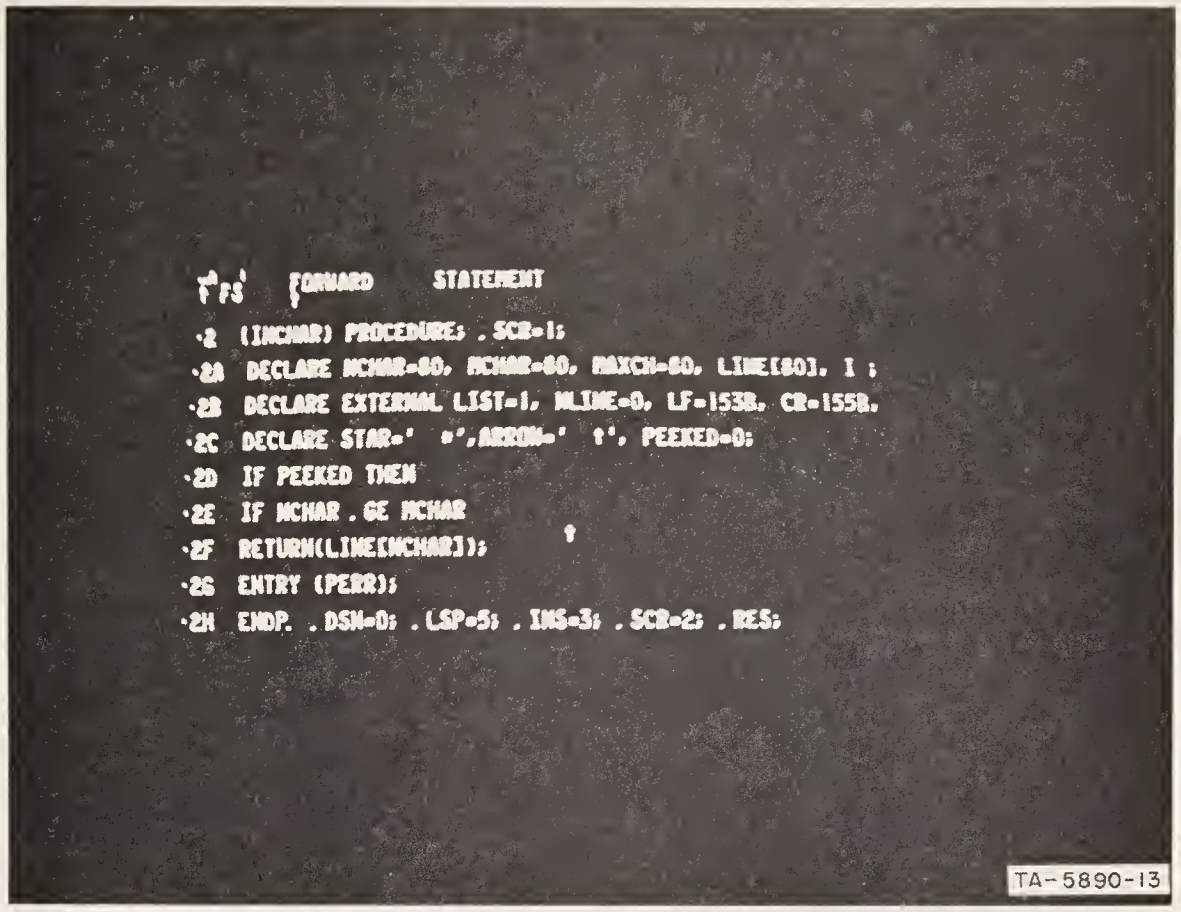


FIG. 3 VIEW OF PART OF AN MOL PROGRAM
Level parameter = 2, truncation parameter = 1

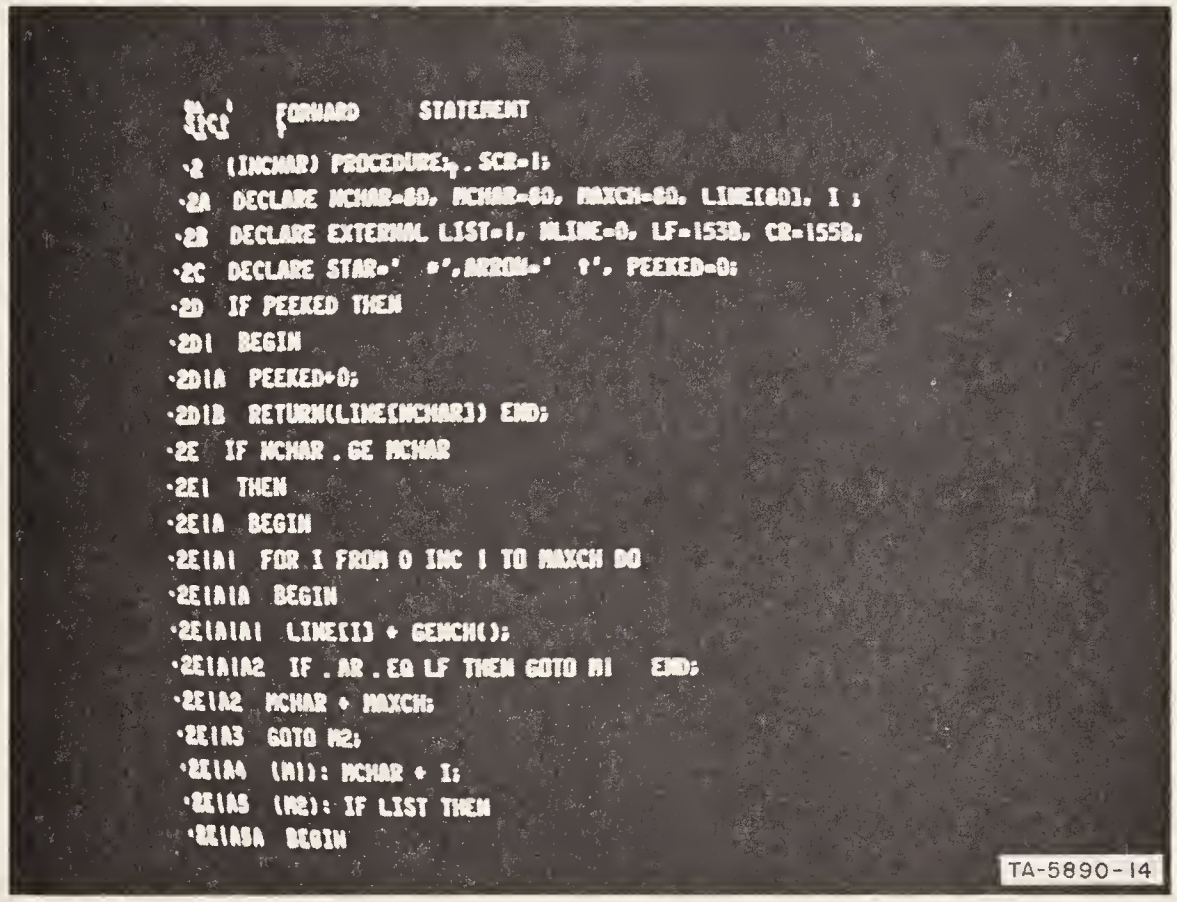


FIG. 4 VIEW OF PART OF AN MOL PROGRAM
Same as Fig. 3, but with level parameter = ALL

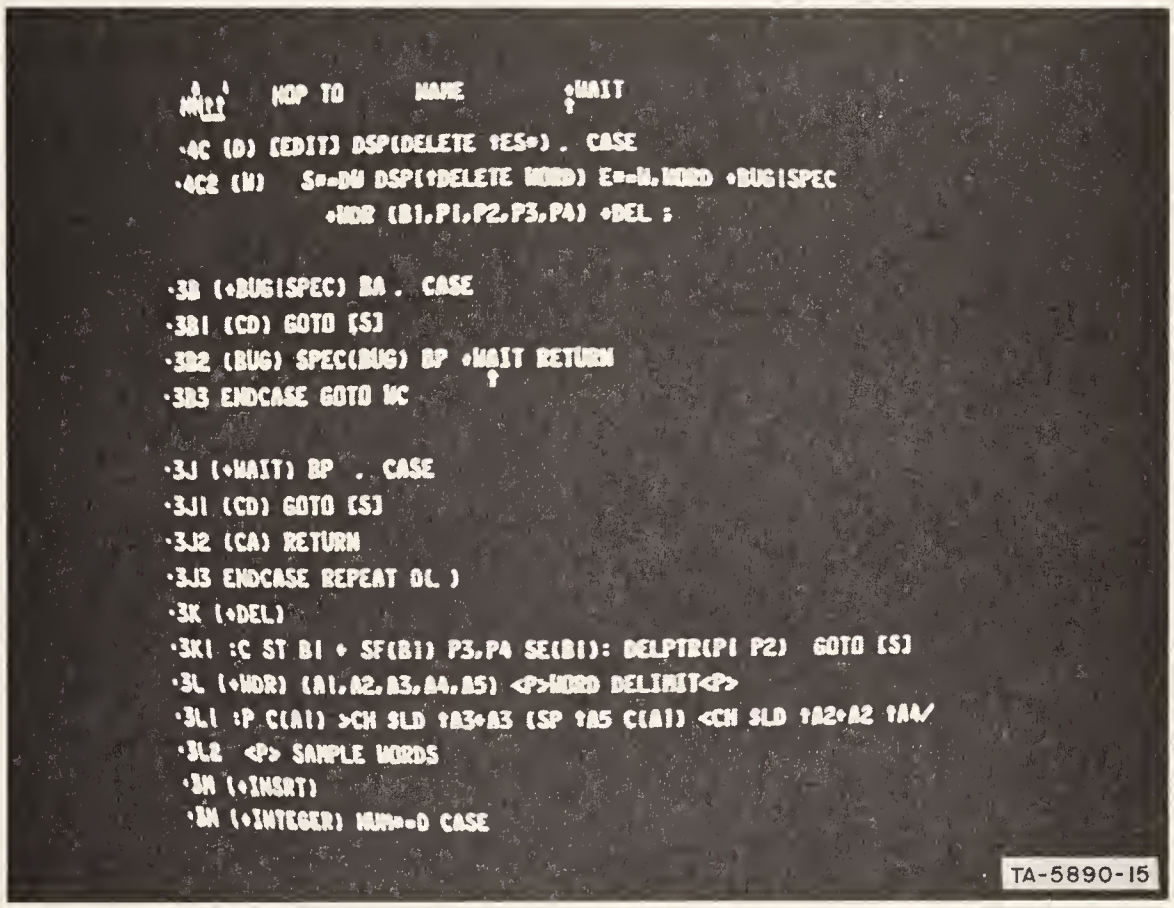


FIG. 5 VIEW OF PART OF A CML PROGRAM, SHOWING USE OF REFERENCE HOPPING. The top six statements are "frozen."

1E2 Structure may also be modified by joining statements or by breaking a statement into two at a specified point.

1F Outputting Files

1F1 Files may be sent to any of a number of different output devices to produce hard copy -- upper/lower-case line printer, on-line high-quality typewriter, paper tape to drive various off-line typewriters, or via magnetic tape to an off-line CRT-to-film system from which we can produce xerox prints, multilith masters, or microform records. NOTE: not all of these are implemented at present.

1F2 Flexible format control may be exercised in this process by means of specially coded directives embedded in the files -- running headers, page numbering, line lengths, line centering, suppression of location numbers, indenting, right justification (hyphenless), etc. are controllable features.

1G Compiling and Debugging Programs

1G1 Source-code files written in any of our home-built compiler languages (see below), or in the SDS 940 assembly laguage (ARPAS, in which our compiler output is produced) may be compiled under on-line control.

1G2 For debugging, we have made a trivial modification to the SDS 940's time-sharing monitor so as to operate the DDT loader-debugger from our CRT displays. Though it was designed to operate from a Teletype terminal, this system gains a great deal in speed and power by merely showing with a display the last 26 lines of what would have been on the Teletype output.

1H Calculating

1H1 The same small innovation as mentioned above for DDT enables us to use the CAL system from a display terminal.

2 Off-Line User Systems

2A This is the third stage in the evolution of our Off-Line Text-Manipulation System (FLTS).

2A1 The first stage was the "Z-Code System" (described in Engelbart, 1965, where it was actually used to do the initial writing for the entire report. It was implemented on a CDC 160A and was used solely for original preparation of paper-tape computer-throughput text, where it allowed selective deletion of most-recent typing.

2A2 The second stage was the addition to the above of a processor that permitted the merging with previously established computer-held files of new input containing new text material as well as instructions to change both old and new material. This system enabled the complete development of documents to be carried out with computer aid, where an arbitrary succession of cycles of addition, modification and reorganization could be carried out. This version of FLTS was described in Engelbart and Huddart, 1965 (which was developed in its entirety with the aid of this text-manipulation system).

2B For this third stage, implemented on our CDC 3100, a set of features were specified to extend and improve those of the earlier systems mentioned above.

2B1 The result is a powerful system for computer processing of textual material prepared off-line on any device which produces punched paper tape or cards; input may be direct or via a computer memory device -- i.e., text stored on a disc may be accessed by the system and handled as if it had been read in directly.

2B2 The output of the system is punched paper tape for use on a Teletype, Flexowriter, or Dura automatic typewriter; alternatively the text may be output to a file in computer memory for later use.

2C Basic Features of FLTS

2C1 This off-line system is fully compatible with our CDC 3100 on-line system (NLTS). They both use the same file-structuring conventions and the same hard-copy output processor. Either system can operate on files produced by the other.

2C2 With FLTS, a newly typed entry file may address any prior text, either within itself or within a previously generated file.

2C3 New statements within these existing structures may be inserted simply by typing an appropriate (perhaps interpolative) location number.

2C4 A prior statement together with its entire substructure (this construct is called a "branch") may be moved to any part of the addressed information structure, merely by designating the new location number (perhaps interpolative) for the referenced statement. Its substructure retains the same relative structural position to the referenced statement in its new location.

2C5 A branch may be deleted simply by referring to its location number (as it appears in the printout, regardless of intervening -- but as yet unexecuted -- specifications for structural

modifications which may eventually result in a new location number being assigned to the referenced statement).

2C6 The text content of any statement may be replaced by merely designating its location number, then typing in the replacement text.

2C7 The text content of any statement may be copied, either as a new statement at a newly specified location number, or as material to be appended to the end of an existing statement.

2C8 Newly typed text content may be appended to any designated prior statement. This new text may include any of the editing instructions described below.

2C9 Within any statement, embedded instructions (special sequences of text characters) may be included (or appended, see above) to specify additions or modifications to that statement.

2C9A Using any one of three different means, a point in the text lying between the statement start and the instruction may be specified.

2C9B Then one can either specify a string of characters to be inserted at that point, or specify that all of the characters from that point forward through the instruction are to be deleted.

2C9B1 It is permissible to include, within the string of characters being inserted at the reference point, an instruction to cause material preceding that point to be deleted -- thus effecting a replacement operation.

2C9B2 One means of specifying the reference point is by quoting a string of characters and specifying whether the reference point is just "in front of" or just "in back of" the first occurrence of this string (within the statement).

2C9B3 Another means of designating a reference point is to count lines, words, and characters relative to the instruction.

2C9B4 The third means is to count lines, words, and characters as follows:

2C9B4A Lines forward from the beginning of the statement or backward from the end of the statement

2C9B4B Words forward from the beginning of the line or

backward from the end of the line

2C9B4C Characters forward from the beginning of the word or backward from the end of the word.

2D Special Features

2D1 When addressing several prior files, whose ranges of statement numbers may overlap and thus threaten ambiguity in their referencing, the user may specify that for his current purposes, each of these files is to have a (different) specified sequence of characters prefixed to every one of its location numbers.

2D1A This allows him to establish temporary, unique referencing to each statement of each file, by prefixing that statement's location number with its file's assigned prefix characters.

2D1B He may thus assemble any number of files into one large structure, within which he may move, copy, delete, and insert statements and substructure sections to effect an arbitrary delete/merge/rearrangement.

2D2 A user may identify different intervals of his new input as pertaining to independent FLTS jobs, merely by specifying a unique "sequence number" for each job.

2D2A To switch his attention (i.e., his associated input material) to a different job, he need only specify the associated new sequence number, and begin typing material for the second job.

2D2B When this new input is subsequently processed by FLTS, the computer will initially isolate the various segments of input occurring between successive sequence-specification points.

2D2C Then, to assemble the relevant new input for each job, it will collect those segments that begin with the sequence-specifying number that corresponds to that job (as communicated to FLTS by the computer operator).

2D2D This provision allows a person sitting at his paper tape punching typewriter to shift his input typing back and forth between various tasks with great flexibility.

2E The off-line system will allow direct interaction with the disc files, allowing an off-line user to modify any file, or merge and modify any sets of files, without the bother of so much intermediate

paper tape.

IV DEVELOPMENTS IN SYSTEM-DESIGN TECHNIQUES

1 The techniques described below represent the base of our software design for the multiconsole system.

1A Introduction

1A1 They emerged from our experience with implementing and modifying our earlier on-line systems, and represent in themselves a significant part of our total research achievements.

1A2 The techniques provide considerable improvement in the following areas:

1A2A Speed and flexibility of implementing and modifying user features

1A2B Thinking about the different levels of system design and about the special design considerations at each level

1A2C Communication between system-development workers, and between them and users

1A2D Currency and quality of system documentation.

2 The User's Control Language

2A The service a user gets from the computer may be considered as a set of discrete operations -- i.e., a set of individual "service functions" from among the repertoire that comprise a "service system."

2A1 Examples of service functions are deleting a word, replacing a character, hopping to a name, etc.

2B Associated with each function of this repertoire is a "control-dialogue procedure."

2B1 Part of the dialogue consists of the sequence of meaningful user actions (keystrokes, select actions, etc.) -- this involves the following:

2B1A Identifying the desired service function

2B1B Setting up the necessary parameter designations

2B1C Recovering from mistakes

2B1D Calling for the execution of the function.

2B2 The other part of the dialogue, interspersed with these user

actions, is made up of computer-feedback messages that help the user in proceeding through these actions.

2C We consider this repertoire of service functions, together with their control-dialogue procedures, to be the user's "control language." This is a language for a "master-slave" dialogue, enabling the master to control application of the slave's capabilities to his own service.

2C1 It seems clear that significant augmentation of one's intellectual effectiveness from the harnessing of computer services will require development of a wide and sophisticated control-language vocabulary.

2C2 It follows that the evolution of such a control language is a very important part of augmentation-system research.

2D For the designer of user systems, it is important to have means for specifying the nature of the functions and their respective control-dialogue procedures, so that a design specification will be:

2D1 Concise, so that its essential features are more easily seen

2D2 Unambiguous, so that all relevant questions about the design may be answered in a straightforward manner

2D3 Canonical, so that certain kinds of information can be found easily

2D4 Natural, so that the form of the description fits the conceptual frame of the design

2D5 Easy to compose, study, and modify, so that the process of evolutionary design can be facilitated.

2E For the user of such a repertoire, it is important to have a description of the nature of the service functions and of their associated control-dialogue procedures.

2E1 The description must be concise, unambiguous, canonical, and natural, for the reasons given above; furthermore, it must be accurate, in that everything relevant to the user about the service functions and their control-dialogue procedures is described, and everything described actually works as indicated.

3 State-Chart Representation of Control-Language Design

3A Figure 6 shows a charting method that was used in earlier stages of our work for designing and specifying the control-dialogue

procedure portions of our control language. Even though limited to describing only the control-dialogue procedures (i.e. not suited to describing the functions to be executed), this representation nonetheless served very well and led us develop the successive techniques described below.

3B Figure 6 shows actual control procedures for four service functions from the repertoire of our current interactive system: Delete Word, Delete Text, Place Up Statement, and Forward Statement. (The procedure flow is described below in the section on "Control Metalanguage.")

3B|1 The boxes contain abbreviated descriptions of relevant display-feedback conditions, representing the intermediate states between successive user actions. Both to illustrate how the charting conventions are used and to give some feeling for the dynamics of our user-system control procedures, we describe briefly both the chart symbols and the associated display-feedback conventions.

3B|A The writing at the top of each box indicates what is to be shown as "command feedback" at the top of the display.

3B|A|1 An uparrow sometimes appears under the first character of one of the words of Command Feedback.

3B|A|1A This indicates to the user that the next character he types will be interpreted as designating a new term to replace that being pointed to -- the absence of an uparrow under Command Feedback signifies that keyboard action will not affect the command designation.

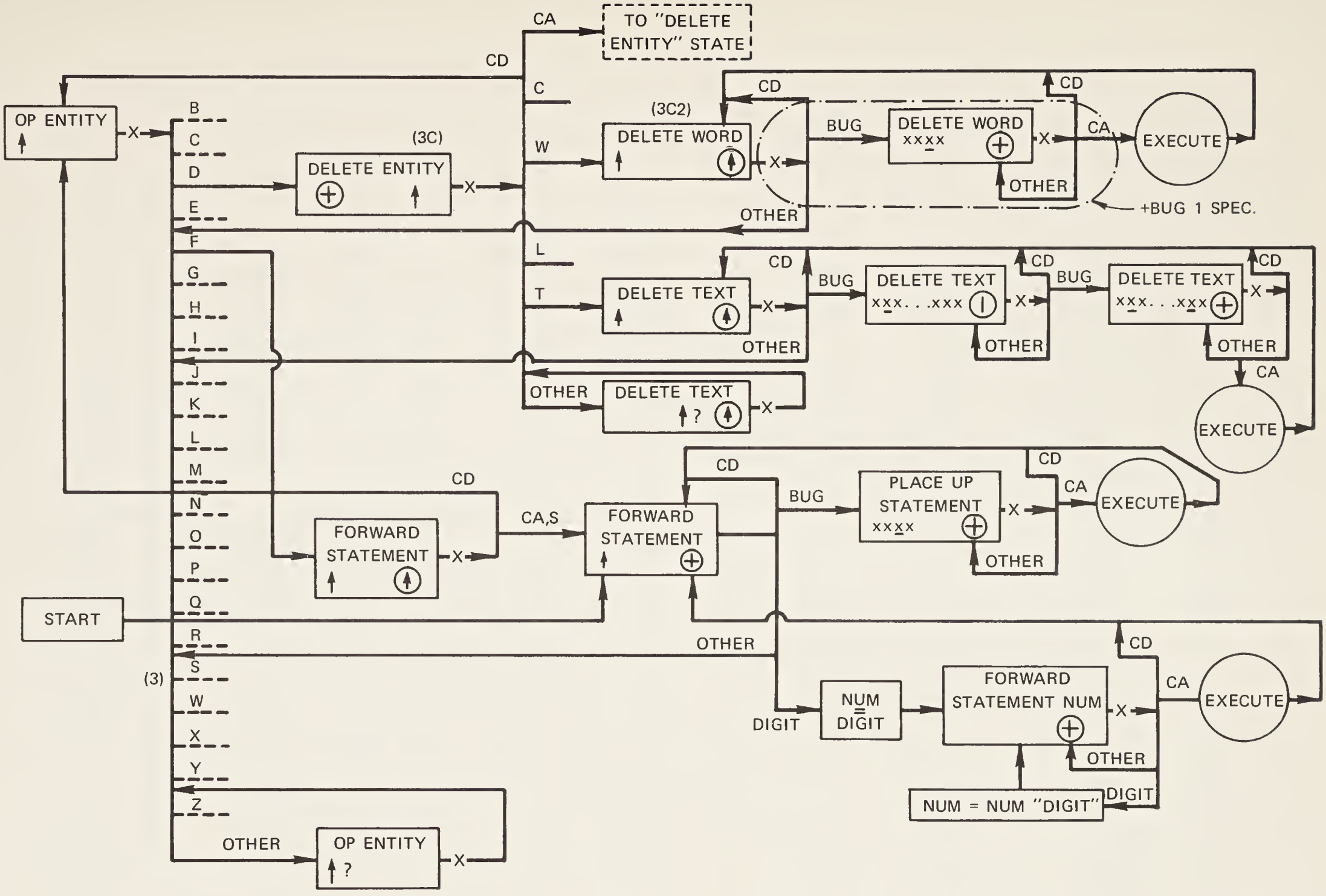
3B|A2 "Entity" represents the entity word (i.e., "character", "word", "statement", etc.) that was last used as part of a fully specified command.

3B|A2A The computer often "offers" the user an entity option.

3B|B The circle in the box indicates the state of the bug (the tracking spot), which alternates between the characters uparrow and plus.

3B|B|1 The uparrow indicates that a select action is appropriate, and the plus indicates that a select action is inappropriate.

3B|C The string of X's, with underlines, indicates that the selected characters are to be underlined as a means of showing



TB-5890-16

FIG. 6 STATE–CHART PORTRAYAL OF PART OF TEXT–MANIPULATION CONTROL LANGUAGE

the user what the computer thinks he has selected.

3B2 Some of the output lines from boxes on the chart are marked with X's. This indicates that the computer is to wait until the user has made another action.

3B2A After this next action, the computer follows a branching path, depending upon what the action was (as indicated on the chart) to reach another state-description box or one of the function-execution processes.

4 Control Metalanguage

4A In search of an improvement over the state chart, we looked for the following special features, as well as the general features listed above:

4A1 A representational form using structured text, so as to harness the power of our on-line text-manipulation techniques for composing, studying, and modifying our designs.

4A2 A form that would allow us to specify the service functions as well as the control-dialogue procedures.

4A3 A form such that a design-description file could be translated by a computer program into the actual implementation of the control language.

4B Using our Tree Meta compiler-compiler (see below), we have developed a next step forward in our means of designing, specifying, implementing and documenting our on-line control languages.

4B1 Figure 7 shows a portion of the description for the control language we currently use, written in "Control Metalanguage."

4B1A This language provides the means for describing both the service functions and their control-dialogue procedures.

4B2 The Control Metalanguage Translator (CMLT) can process a file containing such a description to produce a corresponding version of an interactive system which responds to user actions exactly as described in the file.

4C There is a strong correspondence between the conventions of CML and the state chart, as a comparison of Figs. 7 and 6 will show.

4C1 The particular example printed out for Fig. 7 was chosen because it specifies some of the same procedures shown in Fig. 6.

```
3  (wc:) zap  case

   3A (b) [edit]  dsp(backward ↑es*) .  case
   .
   .
   .

   3B (c) [edit] dsp(copy ↑es*) :s true => <am>adj1: . case

      3B1   (c)  s*=cc dsp(↑copy character) e*=c,character +bug2spec
      +cdlim(b1,p1,p2,p3,p4) +cdlim(b2,p5,p6,p7,p8)
      +cpchtx(b1,p2,p4,p5,p6) ;

      3B2  (w)  s*=cw dsp(↑copy word) e*=w,word +bug2spec
      +wdr2(b1,p1,p2,p3,p4) +wdr2(b2,p5,p6,p7,p8)
      +cpwdvs(b1,p2,p4,p5,p6) ;

      3B3  (l)  s*=cl dsp(↑copy line) e*=l,line +bug2spec
      +ldlim(b1,p1,p2,p3,p4)  +ldlim(b2,p5,p6,p7,p8) :c st b1←sf(b1) p2,
      rif :p p2>p1 cr: then (cr) else (null) ,  p5 p6, p4 se(b1): goto
      [s]

      3B4  (v)  s*=cv dsp(↑copy visible) e*=v,visible +bug2spec
      +vdr2(b1,p1,p2,p3,p4) +vdr2(b2,p5,p6,p7,p8)
      +cpwdvs(b1,p2,p4,p5,p6) ;
      .
      .
      .
      3b10  endcase +caqm ;

3C (d) [edit] dsp(delete ↑es*) . case

   3C1 (c)  s*=dc dsp(↑delete character) e*=c,character +bug1spec
   +cdlim(b1,p1,p2,p3,p4) +del;

   3C2 (w)   s*=dw dsp(↑delete word) e*=w,word +bug1spec +wdr
   (b1,p1,p2,p3,p4) +del ;

   3C3 (l)   s*=dl dsp(↑delete line) e*=l,line +bug1spec...
   .
   .
   .
```

FIG. 7 METALANGUAGE DESCRIPTION OF PART OF CONTROL LANGUAGE

4C2 For instance, the steps of display-feedback states, leading to execution of the "delete word" function, can readily be followed in the state chart.

4C2A The steps are produced by the user typing "d", then "w", then selecting a character in a given word, and then hitting "command accept" (the CA key).

4C2B The corresponding steps are outlined below for the CML description of Fig. 7, progressing from Statement 3, to Statement 3c, to Statement 3c2, to Subroutine +bugspec, etc.

4C2C The points or regions in Fig. 6 corresponding to these statements and subroutines are marked by (3), (3C), (3C2), and (+BUG|SPEC), to help compare the two representations.

4C3 These same steps are indicated in Fig. 7, as starting from Statement 3:

4C3A "D" sets up the state described in Statement 3C

4C3B "W" sets up the state described in Statement 3C2

4C3C The subroutine +bug|spec waits for the select-word (1) and CA (2) actions leading to the execution of the delete-word function.

4C3C1 Then the +cdlim subroutine takes the bug-position parameter and sets pointers P1 through P4 to delimit the word in the text.

4C3C2 Finally, the +del subroutine deletes what the pointers delimit, and then returns to the last-defined state (i.e. to where s*=dw).

5 Basic Organization of the On-Line System (NLS)

5A Figure 8 shows the relationships among the major components of NLS.

5B The Tree Meta Translator is a processor specially designed to produce new translators.

5B1 There is a special language -- the Tree Meta Language -- to use in describing the translator to be produced.

5B2 A special Tree Meta library of subroutines must be used, along with the output of the Tree Meta Translator, to produce a new translator. The same library serves for every translator it

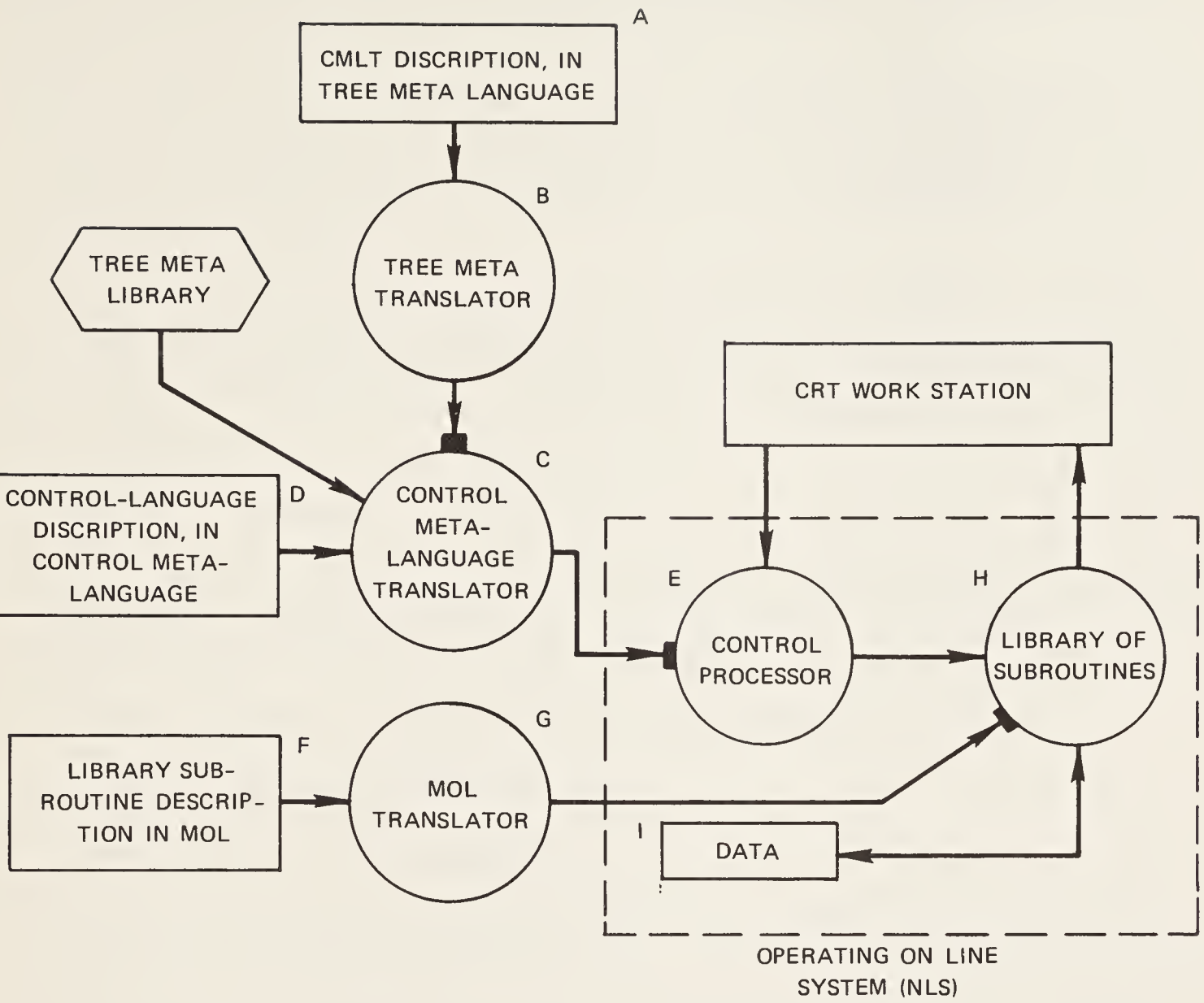


FIG. 8 BASIC ORGANIZATION OF NLS, SHOWING USE OF COMPILERS AND COMPILER-COMPILER FOR IMPLEMENTATION

produces.

5C For programming the subroutines used in our 940 systems, we have developed a special Machine-Oriented Language (MOL), together with an MOL Translator to convert MOL program descriptions into machine code. (SEE Hay, 1968 for a complete description.)

5C1 The MOL is designed to facilitate system programming, by providing a high-level language for iterative, conditional, and arithmetic operations, etc., along with a block structure and conventions for labeling that fit our structured-statement on-line manipulation aids.

5C1A These permit sophisticated computer aid where suitable, and also allow the programmer to switch to machine-level coding (with full access to variables, labels, etc.) where core space, speed, timing, core-mapping arrangements, etc. are critical.

5D The NLS is organized as follows:

5D1 The Control Processor (E) receives and processes successive user actions, and calls upon subroutines in the library (H) to provide it such services as the following:

5D1A Putting display feedback on the screen

5D1B Locating data in the file

5D1C Manipulating working data

5D1D Constructing a display view of specified data according to given viewing parameters, etc.

5D2 The NLS library subroutines (H) are produced from MOL programs (F), as translated by the MOL Translator (G).

5D3 The Control Processor is produced from the control-language description (d), written in Control Metalanguage, as translated by the CMLT (c).

5D4 The CMLT, in turn, is produced from a description (a) written in Tree Meta, as translated by the Tree Meta Translator (b).

6 Advantages of Metalanguage Approach to System Implementation

6A We have means for control-language specification that are much better than before, in terms of being unambiguous, concise, canonical, natural and easy to compose, study, and modify.

6B Moreover, the Control Metalanguage specification promises to provide in itself a users' documentation that is completely accurate, and also has the above desirable characteristics to facilitate study and reference.

6C Modifying the control-dialogue procedures for existing functions, or making a reasonable range of changes or additions to these functions, can often be accomplished solely by additions or changes to the control-language record (in CML).

6C1 With our on-line studying, manipulating, and compiling techniques, system additions or changes at this level can be thought out and implemented (and automatically documented) very quickly.

6D New functions that require basic operations not available through existing subroutines in the NLS library will need to have new subroutines specified and programmed (in MOL), and then will need new terms in CML to permit these new functions to be called upon. This latter requires a change in the record (A), and a new compilation of CMLT by means of the Tree Meta Translator.

6D1 Again, on-line techniques for writing and modifying the MOL source code (F), for executing the compilations, and for debugging the routines, make this part of our work much easier.

V RESULTS AND DISCUSSION

1 Experience with AHI Systems on the CDC 3100

1A Overall Patterns of Usage

1A1 Beginning about October 1966, the number of researchers actively using both text-manipulation systems -- FLTS and NLTS -- has increased considerably.

1A1A This has been especially true since December 1966, when we became almost completely engrossed in the "paperwork" of writing specifications for our new system, and in documenting our old systems. Essentially everyone in the program became involved, and for a period of about six months did all of this work by means of FLTS or NLTS.

1A1B In late Spring of 1967, our staff began to concentrate upon implementation of the 940 hardware and software that we had specified, and their use of the 3100 system fell off considerably.

1A1B1 The software-development work was done with the SDS 940 on-line (Teletype) programming systems which are supplied as manufacturer-maintained subsystems -- i.e., writing/editing (QED), assembler (ARPAS), and loader-debugger (DDT).

1A1B2 With Teletypes in their offices for day-long use, our programmers developed, assembled, debugged, and modified all of their programs on line; thus their files of programs grew and took shape within the internal file structure of the 940.

1A1B3 It would have been too inconvenient and costly to shift these files back and forth between the two computers, to allow the programmers to use the superior composition, editing, and study features of NLTS interspersed with assembling and debugging on the 940.

1A1C FLTS has been used steadily for memoranda, reports, etc., but NLTS has been used over the last eight months of the contract period mostly for peak report-preparation periods and an increasing visitor-demonstration load.

1A2 Observations from Usage Experience

1A2A Appreciation of NLTS, and addiction to it, are generally very quick to appear.

1A2B People naturally prefer NLTS to FLTS, in that the

operations are generally easier to specify and -- more importantly -- are executed immediately.

|A2C However, FLTS has proven very powerful and useful in its supplementary role, since NLTS availability has been very limited (around 20 hours a week, distributed among 12 people during our peak usage in early 1967).

|A2C1 We have come to realize that FLTS would be an important supplement to the research activity on the new system, even though availability of on-line CRT terminals will be much larger (an immediate jump in availability of on-line terminals by a factor of about 20).

|A2C2 FLTS will at least be very useful for enabling clerical staff to transcribe input material effectively; moreover during those inevitable periods when the system is down, users will still be able to be busy specifying additions or changes to their files.

|A2D The set of "tools" provided within NLTS, although quite rich and sophisticated by prior standards of computer-aided text manipulation, represents in our eyes only an initial step toward the type of system that should evolve within several years.

|A2D1 The evolution will be very much accelerated by the fact that we will be able to do essentially all of our work at CRT consoles.

|A2D2 In using the current NLTS we have come to realize that even this "initial" set of tools offers great possibilities for altering various facets of our work, but that it will have to be used for literally hundreds of hours before we can even calibrate the extent and quality of the changes which a thorough integration of these tools into our working methods would produce.

|A3 Problems of Usage

|A3A A dedication to working completely with 3100 systems, while very stimulating and exciting, can produce some very noticeable stresses.

|A3A1 Most of these seem to stem from the very limited availability of NLTS to each person in the program.

|A3A2 Too often, users are trapped into having files stored on the disc which have been recently modified with NLTS, but

for which ther is no hard-copy printout for use in specifying further changes (it is extremely dangerous to use FLTS to specify changes in a file without knowing its current content).

!A3A2A As a result, we often have had to produce hard copy after every NLTS session, since it is very hard to predict whether, during the several days between the first and second usage hours of the week, the user will want to modify the file with FLTS.

!A3A3 Besides the burden of large quantities of printout, it is difficult to coordinate one's work so as to make the best use of the rather tightly scheduled availability of NLTS.

!A3A4 All of these stresses have been reduced very significantly with the much increased on-line availability.

!B An Individual Testimony

!B! Note: Whereas "chopped-up" use of NLTS introduces a certain amount of stress to a user, the experience from the relatively few occasions of spending six or more hours in one working day using NLTS has always proved to be very rewarding and stimulating. Some feelings and thoughts in this regard, as written (on-line with NLTS) by one user (Engelbart) at the end of an eight-hour working session are quoted below.

!B2 "One can almost always do something direct and satisfying about either the big or the little needs that arise:

!B2A "His think-work at hand:

!B2A! "One's effort during any given small interval of time should contribute in some way toward the current working goal.

!B2A!A "As part of our view of a whole day of professional work, we don't really seem to know how to think about the effect of such an interval--what to expect to have learned, to have decided, to have changed in our recorded working information, etc..

!B2A!B "Yet, how to measure, to analyze, and to design (conventions, procedures, computer aids) for the activity of such parcels seems important toward improving the effect of the whole day.

|B2A2 "Being able so easily and quickly to change phrasing and organization within a statement, or to change the structural relationship among statements, somehow gives the feeling of making steady and significant progress toward a desired goal, i.e. of being able to make each small interval count in an explicit way.

|B2B "His miscellaneous notes and thoughts.

|B2B| "To accommodate and preserve a thought or piece of information that isn't related to the work of the moment, one can very quickly and easily insert a note within the structure of a file at such a place that it will neither get in the way nor get lost.

|B2B2 "Later, working in another part of the file, he can almost instantly (e.g. within two seconds) return to the place where he temporarily is storing such notes, to modify or add to any of them.

|B2B3 "As any such miscellaneous thought develops, it is easy (and delightful) to reshape the structure and content of its discussion material.

|B2B4 "It is also easy and delightful to see a number of initially disconnected notes mature to the point where they are ripe to be integrated under one 'topical' heading.

|B3 "This 'being able to do something' about each little notion has profound effects:

|B3A "Perhaps it is a personal problem, but in old-fashioned desk work, or conference work, there always seems to exist for me a large amount of 'mind burdening' stuff that just can't 'have something done about it at the time.' E.g.: notions, thoughts, agreements with others, wondering if ..., a question for so and so when next we talk about ..., etc.

|B3B "In me this builds up a tension that is disagreeable and fatiguing and that also tends to subtract considerably from any end-of-day feeling of accomplishment--there is so much hanging 'in my head' left to be taken care of that it is difficult to relax, to think calmly and deeply, etc....

|B3C "Interruptions of a given work task now needn't represent 'holes in my day's productivity.'

|B3C| "Since in even a short period I can add something to my working structure of information, a feeling that

something was accomplished can be derived from really quite short interruptions about widely ranging matters.

!B3C2 "And when I later shift my attention back to one of these matters, there is a true feeling that those prior short visits can be quite readily salvaged and integrated into the continually developing material.

!B4 "I find that I can express myself better, if I can make all the little changes and experiments with wording and structure as they occur to me." here the user experiments a little with using structural decomposition of a complex sentence.¢

!B4A "I find that I write faster and more freely,

!B4A! "pouring thoughts and trial words onto the screen with much less inhibition,

!B4A2 "finding it easy to repair mistakes or wrong choices

!B4A2A "so while capturing a thought I don't have to inhibit the outpouring of thought and action to do it with particular correctness,

!B4A3 "finding that several trials at the right wording can be done very quickly

!B4A3A "so I can experiment, easily take a look and see how a new version strikes me--and often the first unworried attempt at a way to express something turns out to be satisfactory, or at least to require only minor touch up.

!B4A4 "Finding that where I might otherwise hesitate in search of the right word, I now pour out a succession of potentially appropriate words, leaving them all there while the rest of the statement takes shape. Then I select from among them, or replace them all, or else merely change the list a bit and wait for a later movement of the spirit.

!B4B "I find that,

!B4B! "being much more aware of

!B4B!A "the relationships among the phrases of a sentence,

!B4B!B "among the statements of a list,

|B4B|C "and among the various levels and members of a branch,

|B4B2 "being able

|B4B2A "to view them in different ways,

|B4B2B "to rearrange them easily,

|B4B2C "to experiment with certain special portrayals

|B4B2C| "not available easily in unstructured data

|B4B2C2 "or usable without the CRT display,

|B4B3 "and being aware that

|B4B3A "I can (and am seeking to) develop still further special conventions and computer aids

|B4B3B "to make even more of this available and easy,

|B4B4 "all tend to increase

|B4B4A "my interest and experimentation

|B4B4B "and my conviction that this is but a peek at what is to come soon.

|B4C "And I find that I am more satisfied with the material that I develop.

|B5 "Important to me is that

|B5A "this increased 'practice' with formulating thoughts and relationships,

|B5A| "different in nature from and greater in every-hour quantity than anything I have experienced before,

|B5B "gives me a feeling that it really will increase the growth rate of my intellectual capability."

2 General User Practice and Needs

2A Working Files

2A| About 400 working files are stored on the several disc packs used among the group.

2A2 There has been no significant loss of information due to system failures.

2A2A "User errors" have cost us occasional significant losses. These tend to arise as a result of the high speed with which files can be modified, created, and deleted. Occasionally a user moves too fast, and before realizing the consequences executes some irrevocable "obliteration" operation.

2A2B With growing practice at this kind of fast work, a user tends to use these very features of fast action to create backup copies of files upon which he is currently working.

2A2C This same practice is a standard means of protection against system failures, and partially accounts for the relatively low losses suffered thereby.

2A2C1 A practiced user will still use the system to advantage even under conditions where an intermittent bug may be causing crashes of the system as often as every five or ten minutes.

2A3 The practice of making backup copies is greatly facilitated by having a really quick means for a user to send the current working version of a file out to replace the disc-held version (i.e., to become the new backup version). Total time for such an action is about three seconds with the current system.

2B Practice of Documentation

2B1 During the heavy use of FLTS and NLTS early in 1967, there was a noticeable and generally acknowledged change in the manner in which documentation was accomplished.

2B2 Almost every person was doing most of his work by continually expanding and updating the associated documentation.

2B2A The improved means for expanding and modifying working files encouraged this method of working.

2B2B It was generally agreed that working in this fashion has, in turn, a definite beneficial effect upon the development of projects and the interaction among project personnel.

2B3 There also began to evolve not only a number of special conventions accepted within the group for formatting and handling the files, but also -- more significantly -- a recognition of the scope and complexity of future needs and possibilities in this respect.

2C Structuring of Records

2C1 The generalized hypothesis developed earlier (Engelbart1, 1962 and 1963) about the value of introducing explicit structure to one's working records as a means of deriving more aid from a computer, has been confirmed by experience.

2C1A Indeed, it is likely that there will be a steady increase in the degree and sophistication of explicit structuring within our working records.

2C2 Both FLTS and NLTS derive a considerable degree of effectiveness from the structuring--enough so that none of our practiced users would be willing to forgo these conventions.

2C3 Need for "Parallel" Structure

2C3A It would often be desirable to have more than one substructure for a given statement.

2C3A1 For example, we often find ourselves chopping up a complex statement into a sequence of subsubstatements, as a means of obtaining greater clarity in its expression.

2C3A1A This has a definite appeal as a means of constructing "stronger" structural entities from which to form a structure of description or argument.

2C3A2 At the same time, however, there is often a need for a sublist of "categories" to be included in support of the given statement. If a substructure is already being used to give syntactic clarity to the complex expression represented by the statement, we currently face an awkward problem for which we have no better solution than to join the substatements back into one string.

2C3A3 A very similar situation exists for the "header" statement of a file (i.e., the zeroth-level statement or "null statement").

2C3A3A It is quite useful to consider Statement 0 the "source" statement of the list 1, 2, 3,...

2C3A3B On the other hand, we also need an actual substructure beneath the null statement to carry descriptive information about the file.

2C3A3C Again, we have no ready solution with the current structuring conventions.

2C3A4 In several of the existing formal systems for manipulating list structures, there appears to be a general need for (at least) a double sublist:

2C3A4A One is for the normal list of structural entities, and one is a so called "description list" which serves to describe the parent or source entity "as an entity."

2C3A5 When initially establishing our structuring conventions, we abandoned the possibility of such double substructure in the interest of expediency.

2C3A5A We can always derive an equivalent effect for a given parent statement by giving it two substatements: the substructure of one will serve as a descriptive structure for the parent statement, and the substructure of the other will serve as the "substantive" substructure of the parent statement, but this is a stopgap solution at best.

2C4 With the multiconsole system operative, we would like to search for some ways to meet this need. (The problem is mainly how to display the relationships and how to print the material.)

2D General Problem of Evolving Records

2D1 For the straightforward processes of rearranging and rephrasing that are involved in the evolution of a working file, the direct application of the NLTS command repertoire proves quite satisfactory.

2D2 However, other kinds of "evolution" are going on for which better tools and procedures will be needed.

2D2A When a new feature, process, or concept appears, it is first referred to, perhaps, by some compound-word term; later it may be given a special name.

2D2B Both the terminology and the scope of definition for this newly emerging concept then undergo evolution.

2D2B1 As a change occurs in either of them, a corresponding modification in terminology, organization, and content is often required throughout the associated record.

2D2C Early in the emergence of such a concept, it often becomes apparent that it is likely to evolve in either terminology or definition.

2D2D It seems likely that special tagging given to the term, and associated special identification introduced in the various parts of the working record where there is a dependence upon the definition of the concept, could lead to some useful procedures and computer aids to facilitate subsequent evolution.

2E The Process of Categorization

2E1 This is a very basic operation for developing one's own thinking material, or studying (and possibly integrating into one's own work) the work of another person.

2E1A For instance, the task of developing a useful reference record for the design of a system generally entails the development of categories relevant to the needs of future users of the record.

2E1B Also, in a complex problem, if the subproblems have been properly categorized, one can turn to the solution of certain subproblems with the assurance that this attention to detail will be an efficient investment towards the solution of the larger task.

2E2 We are discovering that the use of NLTS gives us an exciting new ability to interject into the record many detailed notions and notes -- even though they arise while working on an unrelated facet of the problem -- which often leads to finding a large number of miscellaneous notions on one's hands.

2E3 It is easy to place these notions in specified structural locations, which would make it easy to categorize and group them if it were clear what categories were relevant.

2E4 As a representative medium-level task, the matter of establishing a relevant categorization structure, and of integrating miscellaneous types of entities into it, is an important question for study.

2E4A This is a task in which the evolution of an effective repertoire of concepts, conventions, computer aids, and procedural skills will profit significantly from the kind of studies, innovations, and actual usage which we will employ.

3 Control-Language Complexity

3A Although the control language for our present NLTS is fairly sophisticated, a person who becomes skilled with it and who uses it intensively comes to want even more sophistication. Furthermore,

with growth of control-vocabulary size comes a desire for more finesse and efficiency in the associated control procedures to provide more speed and smoothness.

3B Such trends will tend to increase the size and complexity of the control repertoire and to make the encoding of control input increasingly brief. This will mean an increase in the amount of training required to reach peak performance; we regard this as a reasonable price for increased usefulness of the system.

3B1 On the one hand, the control language will always contain a subset which is fairly easy to learn, but which still gives the user the basic means of studying and modifying his files.

3B1A Thus a user will find his vocabulary and skill growing smoothly with his practice and understanding.

3B2 On the other hand, we are trying to discover how valuable such systems can be to a skilled user, and we feel that the decision as to whether or not there is enough value to warrant a given amount of training should be made after we have a set of users who have reached an adequate skill level and thus learned how much value is actually derived.

3B2A From our experience to date, we are certain that for any user to whom a system of computer aids is to provide full-time, interactive working aids, the acceptable threshold on the amount of conceptual and psychomotor training needed for real skill will be a good deal higher than is commonly expected.

3B2B Certainly it will be acceptable if it is as difficult as learning to operate an automobile and to understand and observe the conventions and practices of legal and rational driving rules. This represents a greater set of concepts and skills to be mastered than is generally appreciated.

3B3 Within our own community, a need has developed for formalization of our training procedures.

3C Our Control Metalanguage and "control compiler" approach, as described in Sec. IV-4 for the multiconsole system, should significantly enhance our control-language experimentation.

4 User-Procedure Monitoring and Analysis

4A We developed a modification to NLTS which uses a (spare) magnetic-tape transport to record the sequence of users' actions during an on-line working session. For each recorded action, it identifies the action and gives the time to the nearest hundredth of

a second.

4A1 We then developed an elementary analysis routine to produce usage statistics for specified actions or sequences of actions.

4A2 We did this to answer a specific question relative to system timing, in evaluating the proposed overlay-swap system which was eventually implemented on the 3100.

4B We want to extend this to a significant study, to grow and continue as a regular part of our system study.

4B1 Continuation is being deferred until the multiconsole system is running smoothly.

4C As more people come to be regular users of NLTS, we are observing the following:

4C1 Some users never use some of the features of the system.

4C2 Some execute each operation faster than others.

4C3 Some have developed a better command of the control language, in that they can formulate an effective sequence of on-line operations more quickly.

4D We expect monitoring and analysis to help us to assess these differences objectively and to isolate needs and possibilities for improving the system. For instance, the following activities are planned:

4D1 Direct measurement of the speeds and skills involved in the very basic operations.

4D2 Isolation of the sequences of actions which comprise basic tasks (e.g., assigning a name to a statement, following a reference link, and then returning to the starting point) so that we can study their frequency of occurrence and the particular means by which different users perform these tasks.

4D3 Development of performance characterizations and ratings for these different skills and task sequences, by which we can perhaps develop motivation and/or training.

4D3A In Engelbart (1962 and 1963), a good deal of conceptual work is presented on the nature of one's "system of intellectual capability." The concept of a "hierarchical capability repertoire" is prominent in the discussion of how to improve this system.

4D3A1 It is evident from our experience with NLTS that after continued usage of such a system, the overall capability improvement will arise from many changes which creep into the methods and conventions associated with the capabilities throughout the hierarchy.

4D3A2 For example, the capability for very fast editing and for specifying the content of statements to be displayed are both needed in order to make quick, flexible, and effective use of the search feature.

4D3A3 Speed and flexibility with this searching technique lets one embark more quickly upon searches, and one finds numerous places in reviewing and modifying his files where he begins to use this aid.

4D4 We plan to analyze extensive procedure records so as to bring out the actual hierarchical capability relationships, and not only to witness their evolution but to design for and accelerate that evolution more effectively.

4E It is obvious that such ingredients as the intuitive leaps and heuristic decisions within the behavior of an intellectual worker are of fundamental significance. We are not assuming that the next year of learning how to analyze procedural records will produce the means to isolate and assess this type of high-level ingredient, but we do expect that our work will be an important step toward this sort of achievement -- an achievement which must ultimately be pursued in any serious drive toward augmenting human intellect.

5 On-Line Conference

5A On 12 and 13 October 1967, we experimented with the use of our on-line system of computer aids to support minute-by-minute presentation of information and organization of activities during a two-day progress-review meeting with our project monitors.

5B For this meeting we built a special square table, seating five on a side, with the center area open. We arranged six of our television monitors in the center area, so that each of the 20 persons was conveniently near at least one of them. Figure 9 shows this arrangement.

5B1 The TV monitors were placed low enough to give each participant an unobstructed view of all other participants around the table.

5B2 At one location on the table we set up the mouse, keyboard, and keyset terminal equipment to remotely control NLTS on the CDC

FIG. 9 AN ON-LINE CONFERENCE

3100.

5B3 A TV camera captured the image generated on the large display scope in the computer room, and the video was wired to the six monitors in the conference room.

5B3A The video signals were inverted, so that the displays in the conference room showed black characters on a white background.

5B4 Six auxiliary mice were located around the table, wired so that depressing the control button on any one of them would pick up a relay to connect that mouse into a special input channel to the computer.

5B4A This channel controlled the position on the screen of a second, extra-large tracking arrow.

5B4B This setup allowed any participant to take a nearby mouse and move the pointer about on the screen as a means of calling attention to items on the display.

5C We collected about 150 of the files which have been developed in the program, and put them on one disc pack to be available for access and study during the conference.

5C1 This included all of the work on specifications and some of the symbolic source code of our 940 system design, the documentation and user guides for the 3100 and 940 subsystems, all of our recent reports and proposals, etc.

5C2 We also included some files specially prepared to show the activity, framework, and candidate conference topics, and a framework from which to generate the working agenda.

5C3 Figure 9 shows the specially prepared agenda file with the view parameters set to show only first-level statements, displaying one line of each.

5D This setup proved very valuable as a means for flexibly presenting a large quantity of complex material so that it was easily adjustable to the course of the discussion and to the special needs and questions of the participants. The full power of NLTS gives to a practiced user an unusual capability for presenting his material.

5D1 We are all looking forward not only to regularly running our own program meetings and design sessions in this manner, but perhaps to holding more on-line meetings with sponsors and people outside the program.

5E A photograph and description of this experiment were included in an article by J. C. R. Licklider and R. W. Taylor (1968); Taylor was a participant at our meeting. Their discussion of (and future projections for) computer-aided communication techniques are highly relevant to our work and future developments.

VI CONCLUSIONS AND RECOMMENDATIONS

1 The activity of the project involves a large collection of innovations, implementations, experiments, and analyses.

1A They are all highly interactive, and all in a state that is best described as "initial exploratory."

1B We cannot give conclusive assessments of any of these, but we can give some relative evaluations on various aspects of the program.

2 "Bootstrapping," as a basic research strategy, is beginning to work, and will be much accelerated by the increased on-line service provided by the multiconsole facility.

2A It seems to be paying off in terms of stimulation and orientation.

2B When a person using the system meets a given problem and is also aware of other needs and possibilities arising for users, analyzers, specifiers, and implementers of the system, there is a much increased integration, flexibility, refinement, and speed of innovation.

3 The structuring conventions for our working text are of significant value. Added conventions of this sort will be encouraged.

4 The off-line text-manipulation system has proven very valuable in our past and current operational environment, in cases where on-line time is limited.

4A It seems very likely that such a system could be valuable in writing/publishing activities without having any associated on-line aids.

4B We feel that it will be worth while to implement an off-line system for the new multiconsole system, even though we now have much more display-console availability than before.

5 Our on-line text-manipulation system has now developed to the point where it begins to be significant in the experiment of evaluating what on-line work can really do to augment human intellect.

5A It will be necessary to use NLTS for hundreds of hours per user before a true perspective of its value will develop.

5B Within that period, there is little doubt that the system will evolve considerably further.

6 On-line aids require easy access to be effective.

7 The televised display system developed for our multiconsole system is

economical and flexible. It has several important features:

7A Console cost is very low, since the display is a mass-produced commercial unit (costing about $600).

7B A black-on-white image is optionally possible. This is preferred by users over the usual bright-on-dark CRT display.

7C Video signals from other sources can easily be superimposed on the computer-generated display, to display (for example) a picture of the computer room or of another person's face. The latter should be valuable for cooperative on-line work.

8 There is a very promising field of possibilities relative to new concepts, conventions, techniques, skills, methods, etc. to be developed for users of NLTS (and its descendants).

9 The one-hand keyset, as used within our on-line working environment, is definitely worth the trouble of mastering for those who will devote any appreciable amount of their professional time to working on-line.

9A Extensions toward more keys and more sophisticated encoding are worth pursuing.

10 There will be a large (and complex) body of knowledge and skill to be mastered in order to capitalize effectively on the potential of real-time computer aids.

10A Our experience verifies that an ever more sophisticated interactive language will evolve within our program.

10B This complexity has grown more than was expected. We need to inaugurate a more formal approach to the indoctrination of new people to our conventions and techniques.

11 The features developed to date for our multiconsole system are quite promising, particularly the Control Metalanguage approach to specification, design, and documentation of the Control Language.

12 Our best future research strategy involves the development of a Bootstrap Community for coordinated activities aimed at the most payoff in terms of better augmentation of our own personnel, and better techniques for analysis, design, and implementation of augmentation systems.

13 The development of the Bootstrap Community must be coordinated with the capacity of our consoles, computer service, and file storage to support Community needs, and with our ability to integrate and coordinate people and activities.

REFERENCES

The following entries are in chronological order.

D. C. Engelbart, "Special Considerations of the Individual as a User, Generator, and Retriever of Information," paper presented at Annual Meeting of American Documentation Institute, Berkeley, California (23-27 October 1960).

D. C. Engelbart, "Augmenting Human Intellect: A conceptual Framework," Summary Report, Contract AF 49(638)-1024, SRI Project 3578, Stanford Research Institute, Menlo Park, California (October 1962), AD 289565.

D. C. Engelbart, "A Conceptual Framework for the Augmentation of Man's Intellect," in Vistas in Information Handling, Vol. I, D. W. Howerton and D. C. Weeks, eds. (Spartan Books, Wash)ng4on, D.C., 1963).

D. C. Engelbart, "Augmenting Human Intellect: Experiments, Concepts, and Possibilities," Summary Report, Contract AF 49(638)-1024, SRI Project 3578, Stanford Research Institute, Menlo Park, California (March 1965), AD 640989.

D. C. Engelbart and B. Huddart, "Research on Computer-Augmented Information Management," Technical Report ESD-TDR-65-168, Contract AF 19(628)-4088, Stanford Research Institute, Menlo Park, California (March 1965), AD 622520.

W. K. English, D. C. Engelbart, and B. Huddart, "Computer-Aided Display Control," Final Report, Contract NAS 1-3988, Stanford Research Institute, Menlo Park, California (July 1965).

W. K. English, D. C. Engelbart, and M. L. Berman, "Display-Selection Techniques for Text Manipulation," IEEE Trans. on Human Factors in Electronics, Vol. HFE-8, No. 1, pp. 5-15 (March 1967).

D. C. Engelbart, "Study for the Development of Human Intellect Augmentation Techniques," Interim Progress Report, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (March 1967).

M. Pirtle, "Intercommunication of Processors and Memory," Proc. Fall Joint Computer Conference (November 1967).

J. D. Hopper and L. P. Deutsch, "COPE: An Assembler and On-Line-CRT Debugging System for the CDC 3100," Technical Report 1, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (March 1968).

R. E. Hay and J. F. Rulifson, "MOL940: A Machine-Oriented

ALGOL-Like Language for the SDS 940," Technical Report 2, Contract NAS 1-5904, SRI Project 5890, Stanford Research Institute, Menlo Park, California (March 1968).

D. C. Engelbart, W. K. English, and J. F. Rulifson, "Development of a Multidisplay, Time-Shared Computer Facility and Computer-Augmented Management-System Research," Final Report, Contract AF 30(602)-4103, Stanford Research Institute, Menlo Park, California (April 1968).

J. C. R. Licklider and R. W. Taylor, "The Computer as a Communication Device," Science and Technology (April 1968).